Here is a collection of notes I've made about the various programs that appear on the Program Binaries and Source page. This is intended to give a more full description of each program than can fit in the tables, including what the program does, where it came from, and general thoughts.

als8

This program is courtesy of Paul Schaper. His comments follow:

I used the program quit a bit until I got the Northstar disks. The disk prom was right in the middle of ALS8 and I never tried hard enough to remedy the conflict. The remedy I do have if I really need to run it is that I wrote a virtual Sol program allowing me to run a program from anywhere in memory, but it was too slow to be practical.

ALS-8 is an editor/assembler developed by Processor Technology. My understanding is that there were two versions. One version was loaded from tape, the other version existed on ROMs that came on an S-100 card that plugged into your backplane.

atc

This program is courtesy of Paul Schaper. His comments follow:

This is the best game I ever found for the Sol. In fact, it is the only reason I ever kept the machine this long. I have two versions of the game. One looks like a pre-release version. It has five shortcomings that make it less desirable than the final version:

- 1) The grid is made up of pluses (+) instead of dots (.). It makes it harder to find the airplanes on the screen.
- 2) When you inquire on an airplane, you have to type a question mark after the airplane letter. This doubles the keystrokes to inquire.
- 3) Inquiry does not let you know which direction the plane is traveling. You have to watch it for at least two moves to tell.
- 4) It does not warn you when an airplane is about to enter the airspace. It just surprises you.
- 5) It does not warn you when an airplane wishes to take off from an airport. You have to inquire of each airplane, keep track of those that wish to leave from an airport, and every so often, when the airlane is clear, ask them to take off. They will either do so or say they can't yet.

I do have the docs for this game - a photo copy I believe. And, because I do not have an original tape, I am not sure if I have the right program length.

basic5

This is a 5K BASIC from Processor Technology. It comes courtesy of Paul Schaper. His comments follow:

This tape is labeled: *Processor Technology Corporation; BASIC/5*. It contains:

- BASIC B 0000 19B9
- MTCHS B 1AD9 034D
- LUNAR B 1AD9 0953
- BASIC B 0000 19B6
- MTCHS B 1AD9 034D
- LUNAR B 1AD9 0953

This is the second (of three) basic interpretors PT produced. It is basically the first with fixes. It is fast and the math wasn't bad. The main flaw was that it did not support string variables but it did allow system calls to machine code, with argument passing, that did allow some games to become really spiffy.

basic5 programs

Later Paul contributed a couple other BASIC/5 programs, and figured out a way to allow loading the tokenized BASIC image directly into memory via Solace's "File/Load Program..." menu selection. Normally, this doesn't work because besides loading the program image, BASIC must also twiddle a couple locations that depend on the actual program loaded. Paul simply found those locations in memory and diddled the .ENT file to set those locations as appropriate. This hack wouldn't work on a cassette, but works just fine with Solace. Here are Paul's comments:

Attached is a zip with four Basic5 programs. Lunar and Mtchs are the same from Processor Tech. I modified the beginning so that you can load and run them. In Solace, load and run Basic5. Then load Lunar within Basic5 and run it. The first line in the .ent file (1AD3: FF 3F) causes Basic5 to increase its memory usage from 8K to 16K. It is as if you typed in the command SET M=16384. The second line (1AD7: 2B 24) tells Basic5 where the end of the program is. One nice thing about this way of doing it, you can go from game to game and not have to leave Basic5. Just make sure that Basic5 has stopped interpreting the basic program.

I have also added BOMBR.ENT and BOMB.ENT. Bombr is one of those 101 programs. I included it because I use Basic5's ability to call a machine language routine to poll the keyboard. The routine is near the beginning of the program hidden in a REM statement. Because of this, the beginning of the program is not editable and the whole program is not relocatable (can't move it from 1AD9 to say 2D00).

The second program, Bomb, I saw on a PET computer and got it to work. It uses several machine language routines.

beastie

This is from CPMUG volume 015. A little graphics character performs a random walk, leaving "." as a trail. Not too entertaining for very long.

chase

This is from volume 015 of CPMUG. Vol 029 has an evolution of this version which uses CP/M I/O services. I tweaked it a bit to use the SOL VDM I/O port address.

This is a classic, fun little game. You are stuck in a room which has electrified walls and electrified obstacles filling the room. Making it worse, the room has a number of zombie robots that attempt to catch you. It isn't real-time. For every step you take, they take one step. The goal is to evade them without walking into any electrified obstacles while at the same time trying to lead them to walk into the obstacles. (the robot chase algorithm is simple: take the most direct path).

From the source code:

```
THE GAME OF CHASE
    -- FROM DR. DOBBS JOURNAL
           MAY 1977
    -- MODIFIED FOR SOL
```

chess

This program comes courtesy of Paul Schaper. His comments follow:

This tape is labeled: *Processor Technology; 8080 Chess Cassette; P/N 727152*. It contains:

```
CHESS    0000 16F9
END      CC00 0000
```

A middle of the road chess program at the time. Good for their first effort. It out performed all the other chess programs I had for the Sol (I never got any of the CP/M ones) but Zargon ran circles around PTs chess. It always made a move on level 3 but if it was raised to level 4 and the board became complicated, it would get stuck in an infinite loop (the stack would wrap around itself).

dbg8, dbg16, dbg32, dbg48, dbg64

These programs come courtesy of Paul Schaper. His comments follow:

This is the last tape I have. It is labeled: *Processor Technology; DEBUG, Advanced 8080 Debugger; P/N 727132*. It contains:

```
DBG8    0E00 11D4
DBG16   2E00 11D4
DBG32   6E00 11D4
DBG48   AE00 11D4
DBG64   EE00 11D4
END     CC00 0000
```

I never used it because I don't think I had the docs and I never had a problem sticky enough to need it. One of the versions on the tape was never able to load but I was able to create it by comparing the differences between two others and making the necessary changes to create the bad one.

deflect

This is from CPMUG volume 029. Quoting the CPMUG docs:

```
DEFLECT.ASM ANOTHER VDM GAME. THIS ONE INVOLVES
MORE COORDINATION THAN PIRANHA, BUT
IS MORE SIMPLE-MINDED IN ITS PLOT.
```

The source code says:

```
THE GAME OF DEFLECTION BY ANDREW A. RECUPERO
FROM KILOBAUD FEB '78 #14
```

I had to tweak the I/O ports to work for the SOL. Directions are in the source code.

edit, unpac, pack

These programs come courtesy of Paul Schaper. His comments follow:

This tape is labeled: *Processor Technology; EDIT, Advanced 8080 Editor; P/N 727142* It contains:

```
EDIT     0000 1A44
UNPAC    0000 03E0
PACK     0000 0364
END      CC00 0000
```

I never used these programs because I never saw the docs and I had other programs to do the work.

extbasic

This program comes courtesy of Paul Schaper. His comments follow:

This tape is labeled: *Processor Technology; Extended Cassette BASIC; P/N 727019*. It contains:

```
BASIC B 0000 3F85
END   ? FFFF 0001
BASIC B 0000 3F85
END   ? FFFF 0001
```

I rarely used this program because it was too big to use with my tape player, it required me to relocate my Northstar Dos to a place I didn't want it, it was slow, and the math package wasn't that strong.

I present two versions of the program, which are identically long:

- extbasic.ent
- extbasic_patched.ent

extbasic.ent is the pristine original source. It has a bug where:

```
10 FOR I=1 TO 0
20 FOR K=1 TO 0
30 PRINT "impossible"
40 NEXT K
50 NEXT I
```

will cause a CS (control stack) error. P.T. put out an errata sheet with a small patch to fix this bug. That has been applied and is in the file extbasic_patched.ent. This indeed fixes this problem. However, when running "ROMLN.BAS" (which uses a lot of `NEXT 100` type statements and fiddles with the control index), we get a CS error with the patched version but not the original version.

focal

This program comes courtesy of Paul Schaper. His comments follow:

This tape is labeled: *Processor Technology; 8080 Cassette Focal(tm); P/N 727027*. It contains:

```
    FOCAL     0000 1C93
    END       CC00 0000
```

I wrote three programs in this language to gauge its ability. It used binary math rather than BCD (binary coded decimal) so the math errors and lack of capability made it unuseful.

To drive this last point home, Paul gives the example:

```
*T 23+31
 54.0001
```

Now that's really bad!

The program was written by Robert Arnstein and was portable to other 8080 systems. This program source doesn't exactly match the source listing provided by Processor Tech, but it is close and has the benefit of being ASCII. For that matter, the pdf'd source listing doesn't exactly match the execuable distributed as Cassette Focal 1.0 either. Arnstein's comments indicate it was written in 1975; the pdf'd source listing indicates PT distributed it in 1976; the executable title screen is copyright 1978.

It is somewhat intersting that the program uses floating point routines developed for the 8008 microprocessor in 1973, then ported to the 8080 in in 1975.

galaxy

This game was reconstructed from the source listing in Processor Tech *Access* newsletter, Volume 1, Number 4. It is a logic puzzle where you try to "shoot stars" (flip bits) to change the starting pattern to the goal pattern. The instructions are given when the game is played. Looking at the source might also provide some clues.

memtest

Philip Lord has a CDC 64KB memory card in his Sol. He modified the memory test to allow checking all 64KB. The zip file contains the manual for the CDC memory card as well as .ENT files for the memor test.

microchess

This is a classic, early chess program for microcomputers, written by Peter Jennings. It was ported to many platforms.

Moves are entered using decimal coordinates, using this grid:

```
00 01 02 03 04 05 06 07
10 11 12 13 14 15 16 17
20 21 22 23 24 25 26 27
30 31 32 33 34 35 36 37
40 41 42 43 44 45 46 47
50 51 52 53 54 55 56 57
60 61 62 63 64 65 66 67
70 71 72 73 74 75 76 77
```

A typical opening move might be entered as 64-44

This program was recovered from a cassette tape donated by Bob Senzig, in memory of his brother, Don Senzig, Jr., of Milwaukee, Wisconsin.

mine

This game is courtesy of Paul Schaper. His comments follow:

This tape is not from PT. I bought it from a catalog I found. I may still have the catalog. The tape is simply labeled: *MINEFIELD*. It contains:

```
MINE  G 0000 1495
MINE  G 0000 1495
NOMOR ? FFFF 0001
MINE  G 0000 1495
```

It is a game. Try to get from one end to the other of a mine field.

When you run the game, the opening screen supplies directions.

music

This program and associated "score" files come courtesy of Paul Schaper. His comments follow:

This tape is labeled: *MUSIC SYSTEM*. It appears to by produced by Software Technology Corp, a division of Processor Technology Corp. It contains:

```
MUSIC M 0000 082E
PRELD M 08D3 09AE
ALLEG M 08D3 02D2
SARAB M 08D3 059F
AIR   M 08D3 0CF7
CHORL M 08D3 08B4
END   ? 0000 0001
```

This is the music program that caused the disappearance of CPMUG disk 39. It toggles the interrupt enable line on the CPU on and off at a fast rate. The Sol did not use this line. I tried to get it to run of a Northstar (Z80) but the hardware was too different. Three voice harmony. The price was right considering any other method of making music cost over $100 per voice.

Until Solace has support for emulating the filtered interrupt enable output and producing sound, there isn't a whole lot of point in running this software. Someday...

piranha

This is from CPMUG volume 029. Quoting the CPMUG docs:

```
PIRANHA.ASM A SUPER VDM GAME FROM INTERFACE AGE.
JUST CHECK THE I/O ROUTINES, ASSEMBLE,
AND HAVE FUN DODGING THE HUNGRY FISH.
```

I had to tweak some of the port address info for the SOL.

A charming little game. Little asterisk piranhas swim around and if they run into you, you die. The longer you live, the higher your score. You can even redefine the key mappings. The fish have different behaviors: sometimes swarming, sometimes just swimming past. I don't have all the controls figure out.

By default, ESC sets up key mapping:

- CTRL-A goes back to the monitor
- 8 is up, 4 is left, 6 is right, 2 is down
- 7 is UL, 9 is UR, 1 is DL, 3 is DR
- 5 is stop moving
- . is pause the play
- 0 is AUTO, although I don't know what this does

If you hit a key twice, it goes that way quickly.

There is also a PHASE counter at the bottom, but I'm not sure how that works either.

raiders

This is a clone of Space Invaders, written by Steve Maguire. Steve is the same incredibly generous guy who gave me two Sol systems and a bunch of software. He is also the author of a some pretty widely read books on programming and project management, including Writing Solid Code and Debugging the Development Process.

robot

This is from CPMUG volume 029. Quoting the CPMUG docs:

```
ROBOT.ASM LET YOUR VDM MOONMAN BUMP AROUND
THE BOUNDARIES OF THE SCREEN.
```

Source code comments say "FROM BYTE, APRIL 1978". It is a somewhat pointless program. A little man does a random walk around the screen.

I had to tweak the I/O ports for the VDM and keyboard. Typing CTRL-C reboots the computer.

space

These programs come from a "Space Games" tape supplied by Bob Stek. They were read and dumped twice and verified against each other.

Here is what the headers contained, and this is the order they appeared on tape:

```
ASTER   3000 0901
LUNAR C 3C20 1FFC
STRWR C 3C20 144D
ROMLN C 3C20 13F4
```

LUNAR, STRWR, and ROMLN are all tokenized (binary) Extended Cassette BASIC programs. The *.ECB versions of the programs were generated via a perl script by detokenizing the binary program.

strategy

These programs come from a "Strategy Games" tape supplied by Bob Stek. They were read and dumped twice and verified against each other.

Here is what the headers contained, and this is the order they appeared on tape:

```
TRAP    3000 0901
WMPUS C 3C20 0E9B
WMPS2 C 3C20 254C
RACE  C 3C20 227D
KNGDM C 3C20 1438
```

WMPUS, WMPS2, RACE, and KNGDM are all tokenized (binary) Extended Cassette BASIC programs. The *.ECB versions of the programs were generated via a perl script by detokenizing the binary program.

targ

This is the program that made the Sol famous. It is quite impressive that such a good game can be made from ASCII graphics. Although the "spash screen" doesn't pop up the following message, a binary dump of the program reveals the following revision notice:

```
TARGET
COPYRIGHT 1977
BY
PROCESSOR TECHNOLOGY CORPORATION
VERSION 2.4   JANUARY 7, 1977   S. DOMPIER
```

This game was also ported to run on the TRS-80, which also happens to have a 64x16 memory mapped display.

target

This is from volume 015 of CPMUG. Volume 029 of CPMUG contains an evolution of this program that uses CP/M services for I/O. There are no source code credits, but the CPMUG directory listing gives credit to "GEORGE W. ROMPOT".

This is not the same thing as the more famous target game that went by the name "TARG". In this game, your player is in a fixed location on the left side of the screen and you attempt to shoot a bouncing object on the right side of the screen. It is quite inane.

train

It is a screen saver, of sorts. An ascii graphic train chuffs across the screen. For the full effect, the dipswitch under the keyboard set to display the right set of chars.

I have three versions of this program, but present only one. If you think you might want one of the others, let me know and I can send it along.

The version I present is the source as scanned from the Processor Tech *Access* newsletter, Vol 1, Number 4. There was also a version presented in Vol 1, Number 1, but it couldn't run on the Sol; it was written for some other S-100 system using the VDM-1. I also have a version of this from volume 001 of CPMUG, but the source listing appears to be a reconstruction of the source by someone who typed it in from the Access newsletter, but didn't want to type in all the comments.

The *Access* newsletter says the author is "Newett Awl," but I'm not sure this isn't actually a pun, you know, like "know it all." I've heard that Newett Awl was actually Gordon French.

winzi

This program was reconstructed from the listing appearing in P.T. *Access* newsletter, Volume 1, Number 4. It is a tiny program that contains four very terse "games." One neat feature is that these all run entirely from the SRAM on the Sol motherboard, so that a bare system without any memory expansion can still play the games.

The four games are Chase, Hic, Opps, and Life. See the newsletter for more details on the program. Don't forget to `SET S=80` from the monitor before playing the game, otherwise the game speed is way to fast.

Contact: jim@thebattles.net (Jim Battle)

Last update: July 10, 2011